

Different Plugin API Versions

OpenCPN 3.1.x is downward compatible with all previously built and/or distributed Plugins:

- Inversely, older Plugins are upward compatible with new versions of OpenCPN.
- Except when there is a change in the wxWidgets version.
- When there is a change in the wxWidgets version, all plugins must be recompiled using the new wxWidgets version.
- The minimum plugin API number that is required for a given plugin is shown in the Plugin API in the source code (see below).
- The plugin API is contained in the *opencpn.lib* file.
- Upward compatibility is a fixed design goal (with the exception of wxWidgets changes).
 - But, the opposite is not true. Plugins built to 3.1.x API spec are not compatible with OpenCPN 3.0.x.
 - And the corollary: OpenCPN 3.0.x is not compatible with Plugins built to the 3.1.x API spec.
 - This second case should be clear, logically. How could OCPN 3.0.x anticipate what the the 3.1.x PlugIn API extensions would look like?

So, as a Plugin Developer, the model to follow is:

- If you want a durable Plugin that runs on 3.0.x and runs on all subsequent versions, code to the 3.0.x API.
- ...unless...you want the extended features provided by the 3.1.x PlugIn API, in which case you should use the 3.1.x API to create an Enhanced Plugin, which will be compatible with 3.1.x and all versions to follow.

Change in wxWidgets versions & PI Compatibility

1. wxWidgets 2.8 libraries were used for Opencpn versions up to 4.0, requiring **all** plugins be updated.
2. wxWidgets 3.0.2 libraries were used for OpenCPN version 4.1 through 4.8.8, requiring **all** plugins be updated for compatibility with this new library, which of course has reset the compatibility bar, and made those new plugins only forward compatible from version 4.1.
3. wxWidgets 3.1.1 libraries are used for OpenCPN version 4.99.xx and up requiring **all** plugins be updated.

Plugin API Designation

Plugin API Designation found in the `github/[plugin]_pi/src/[pluginname].h` file:

```
#define MY_API_VERSION_MAJOR 1
#define MY_API_VERSION_MINOR 13
```

Plugin File Name Format

lib_ [PluginName]_ [OSdesignation]_pi [Plugin-API]_v [VersionNumber]_pi.dll

eg: **lib_aisradar_win32_pi18_v32_pi.dll**

eg: **lib_aisradar_win32_pi.dll** (Simpler version) eg: **lib_aisradar_win32_pi18_v32_pi.exe** (as an executable)

1. **lib** is the designation for a released plugin. Beta Versions of plugins will not have the [LIB] prefix.
2. **PluginName** is the name of the plugin.
3. **OSdesignation** is Operating system designation "win32" or "win64", etc.
4. **Plugin-API** is the Plugin API number required by this PlugIn "18", "19", "10", etc. which also equates to designated main OCPN program API numbers. See chart below.
5. **VersionNumber** is the private version number of the Plugin itself, controlled by the Plugin developer.
6. **.dll** Means that the plugin is a Windows version of the plugin, probably compiled using MSVC

Plugin (pi) API Versions and OpenCPN Version Compatibility

In plugin filenames there generally be will be a designation as above (it is part of the plugin filename format).

It tells which API version is needed and, if it was built against the right codebase, with which version it should run - Startup of Opencpn leaves messages in the opencpn.log regarding the status of each plugin. If the plugin is not installed because the current Opencpn version does not support it, the piXXX designation will help you decipher what the problem is and what can be done.

pi18 (1.8) = OpenCPN 3.0 [Use oldest supported Opencpn 3.2.2]

pi19 (1.9) = OpenCPN 3.2 [Use oldest supported Opencpn 3.2.2]

pi19 (1.9) = OpenCPN 3.2.2 [Oldest supported version]

pi110 (1.10) = OpenCPN 3.3.x

pi110 (1.10) = OpenCPN 3.4 <—wxWidgets 2.8.1 used

pi111 (1.11) = OpenCPN 3.4 <—wxWidgets 2.8.1 used

pi112 (1.12) = OpenCPN 3.4 <—wxWidgets 2.8.1 used

pi112 (1.12) = OpenCPN 4.0 <—wxWidgets 2.8.1 used

pi113 (1.13) = OpenCPN 4.1 <—**wxWidgets 3.0.2 first used**

- Plugins must be compiled using wxWidgets 3.0.2 to be compatible.

pi113 (1.13) = OpenCPN 4.2 <—SVG Icons added (gtk,expat,vc,etc)

pi114 (1.14) = OpenCPN 4.5.307 <—Externalize AIS drwg #795 -Rel.Candidate

pi114 (1.14) = OpenCPN 4.8.0 See Note just below, when API 1.14 was fixed pi114 (1.14) = OpenCPN 4.8.0 Changed API 1.14 to work in MS Windows for BR24beta

- Externalize AIS drawing #869 8/10/2017 .
- API 1.14 fully working in OpenCPN 4.8.0.
- In the broken OpenCPN 4.6.0 implementation nobody was using that feature of the API until it was

fixed.

pi115 (1.15) = OpenCPN 4.8.2

- Extend with functions providing min/max available basemap quality #974
- Fix Grib plugin to use same screen size as main application #893
- Implement api call for plugin to handle autopilot routing #996
- Correct route & track messages so they can actually be used by plugins #995

pi116 (1.16) = OpenCPN 4.99 (5.00 or ov50) <—**wxWidgets 3.1.2 first used**

- Plugins must be compiled using wxWidgets 3.1.2 to be compatible.

pi117 (1.17) = OpenCPN 5.x with wxWidgets 3.1.2, new Plugin Manager & Versioning

- See [Cruiser Forum Post 2502](#)
- Adds extern DECL_EXP bool ShuttingDown(void);
- Adds extern DECL_EXP wxString GetPluginDataDir(const char* plugin_name);
- Found in github.com/Opencpn/include

Available as compressed download

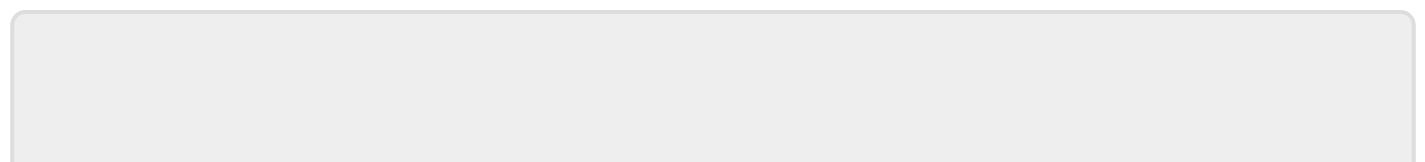
- [Opencpn-lib.zip](#)
- [Opencpn-lib.7z](#)
- [Sourceforge: opencpnplugins Windows Opencpn Development Libraries/](#)
- [Sourceforge opencpnplugins Opencpn_Lib](#)

Sourceforge files are not as complete.

Head Developer: About Plugins & API

The Main program API number is distinct and independent from the OpenCPN Version number. Each released version of OCPN will support a certain PlugIn API. As we progress, we expect the API eventually to stabilize, or at least to evolve more slowly than the main program. This will happen naturally as we fill out the API to include the commonly required Plugin functions.

I think it would be a great idea to bring some order to the Plugin constellation. There may be some renaming of existing Plugins required, but that's OK at this point. We definitely ask all Plugin developers to try to follow the naming convention during their development, and especially after their release. One can see at a glance what the requirements of a particular Plugin will be.



From:
<https://opencpn.org/wiki/dokuwiki/> - **OpenCPN Manuals**

Permanent link:
https://opencpn.org/wiki/dokuwiki/doku.php?id=opencpn:developer_manual:plugins:plugin_api:plugin_api_versions

Last update: **2020/05/26 12:01**

